applied sciences

MDPI

Article

Logic Synthesis Strategy Oriented to Low Power Optimization

Marcin Kubica [1,*], Adam Opara [2] and Dariusz Kania [1]

[1] Department of Digital Systems, Silesian University of Technology, ul. Akademicka 2A, 44-100 Gliwice, Poland; dkania@polsl.pl
[2] Department of Graphics, Computer Vision and Digital Systems, Silesian University of Technology, ul. Akademicka 2A, 44-100 Gliwice, Poland; aopara@polsl.pl
* Correspondence: mkubica@polsl.pl

Abstract: The article presents a synthesis strategy focused on low power implementations of combinatorial circuits in an array-type FPGA structure. Logic functions are described by means of BDD. A new form of the SWitch activity BDD diagram (SWBDD) is proposed, which enables a function decomposition to minimize the switching activity of circuits. The essence of the proposed idea lies in the proper ordering of the variables and cutting the diagram, ensuring the minimization of switching in the combination circuit. This article contains the results of experiments confirming the effectiveness of the developed concept of decomposition. They were performed on popular benchmarks using academic and commercial synthesis systems.

Keywords: low power synthesis; FPGA; switching activity; decomposition; technology mapping

Citation: Kubica, M.; Opara, A.; Kania, D. Logic Synthesis Strategy Oriented to Low Power Optimization. *Appl. Sci.* **2021**, *11*, 8797. https://doi.org/10.3390/app11198797

Academic Editor: Alfio Dario Grasso

Received: 25 August 2021
Accepted: 20 September 2021
Published: 22 September 2021

1. Introduction

Logic synthesis aimed at minimizing power consumption is currently one of the most important problems in the design of energy-saving digital circuits. One of the most popular families of digital circuits is the very complex programmable Systems on Chips using LUT-based FPGAs. Until recently, the goal of optimizing the synthesis process was to minimize the number of LUTs or to minimize the length of the so-called critical path affecting the maximum clock frequency. Along with technological development, the problem of heat dissipation began to play a dominant role; thus, resulting in the need to reduce power consumption. The development of mobile devices is also important, as it forces the necessity to minimize energy consumption.

Minimizing power consumption should be sought at every level of circuit design. The stage of the system synthesis, at which the greatest limitations of power consumption can be found, undoubtedly depends on the specificity of the designed system. The greatest benefits are to be expected when looking for a reduction in power losses at the highest design levels [1]. The decision to divide the task into software and hardware parts can have a significant impact on energy efficiency [2]. Only at the highest design level is it also possible to anticipate the temporary shutdown of the modules of the designed hardware and software system, which undoubtedly significantly reduces the total power of losses. Many concepts of power minimization at the system level can be found in [3–6]. They are often related to the problems of high-level synthesis, presented, inter alia, in [7–10].

However, the power savings at lower synthesis levels are also important. From the point of view of saving power consumption, the most advantageous techniques are minimization techniques aimed at minimizing both static and dynamic power consumption. Such a concept of reducing power consumption are undoubtedly the methods of local lowering the supply voltage [11,12], or recently very popular methods called "power gating" [10–14]. However, they cannot be used in FPGAs. Methods aimed at minimizing dynamic power consumption, with particular emphasis on automatic logical synthesis algorithms, are also of great importance in the minimization process. Most of them are

related to the methods of limiting the power consumption of synchronous systems. The essence of many methods of synthesizing energy-saving circuits comes down to minimizing the switching. Among them, there are various techniques for limiting the frequency of the clock signal. Sometimes it is possible to reduce the frequency of the global clock signal, but this leads to a longer execution of individual functions, which often adversely affects the operation of the entire system. In this situation, the possibility of temporarily blocking the clock signal (clock gating) [15] is often sought. The possibility of blocking the clock signal can be predicted by analyzing the state of the input signals. There are many architectural concepts known for the possibility of blocking the clock signal. Among them, for example, it is worth distinguishing the precomputation logic method [16], the kernel extraction method [17–19], or various FSM decomposition methods [20,21].

Much more sophisticated solutions are related to the local lowering of the clock signal frequency, which are possible to be implemented in less loaded modules of a complex digital system. In this case, however, there is a problem with the synchronization of data exchange in two modules working with different clock signals. A very interesting concept of reducing power losses is the implementation of systems in the form of a GALS (globally asynchronous locally synchronous) structure, which makes it possible to adjust the frequency of the local clock signal to the computational needs of a given unit, but requires the use of asynchronous interface blocks enabling data exchange. Solutions of this type work well in complex systems in which the modules are not evenly loaded with the performed calculations [22]. It should be emphasized that methods of limiting power consumption begin to play a key role in low-level synthesis. Appropriately carried out selected synthesis steps can lead to a reduction in power consumption by programmable systems. This is especially visible in the case of the implementation of FSMs [23–25]. There are several ways to reduce power consumption when implementing FSM. The methods associated with the appropriate FSM internal state coding [26] are very promising. Moreover, in the case of combinational systems, there are methods that enable a reduction in power consumption, as shown in the works of [27–31].

The aim of this article is to present a low-level method of logical synthesis leading to a reduction in power consumption by combinational circuits implemented in FPGAs. This method is directly related to the function decomposition and technological mapping in LUT-based FPGAs. Appropriate modification of the decomposition leads to the creation of a new synthesis strategy aimed at minimizing the power consumption of the logical structure.

The first part of this article presents the theoretical basis of decomposition using the description of a function in the form of BDD (in its reduced ordered form ROBDD). Then, the essence of limiting power consumption by reducing switching activity is presented. In the next part of the article, a new form of the diagram is introduced—SWBDD—which allows the authors to propose an appropriate synthesis strategy aimed at minimizing dynamic power consumption. The next section contains a general description of the algorithm implemented in the PowerDekBDD tool, in which the ideas presented were implemented. The article ends with the section "Experimental Results," presenting the effectiveness of the described strategy, and "Summary," which contains conclusions and an indicated direction for further work.

2. Theoretical Background

One of the main tasks of logic synthesis geared toward FPGAs is to map the implemented logic structure into FPGA resources. The technology mapping in the case of combinational circuits comes down to the mapping of the implemented function from LUT logical blocks included in the circuit. It turns out that LUT logical blocks are capable of executing any logic function with a limited number of variables. The number of k inputs of the LUT (number of function variables) is small. Therefore, there is a problem of dividing the implemented circuit. The mathematical model of such a division is the decomposition of a function. The decomposition theorem was formulated by Ashenhurst [32] and developed by Curtis [33] as early as the 1960s. In the classical approach, simple serial

decomposition comes down to splitting all variables into two sets: bound set *Xb* and free set *Xf*. The variables contained in the bound set are variables for the bound functions *g* [33], which are implemented in the bound block. On the other hand, the bound functions and the variables from the free set are variables of the free function realized in the free block. From the point of view of efficient partitioning, the key is the number of bound functions *p*, which corresponds to the number of wires between the bound and free blocks.

Let *f* be n-input and m-output logic function reflecting B^n set into B^m set i.e., f: $B^n \rightarrow B^m$, where $B = \{0, 1\}$. Function f: $B^n \rightarrow B^m$ may be presented as $Y = f(I_{n-1}, \ldots, I_1, I_0)$, where $Y = \{y_{m-1}, \ldots, y_1, y_0\}$.

Function f: $B^n \rightarrow B^m$ is subjected to decomposition, i.e.,

$$f(X_2, X_1) = F[g_1(X_b), g_2(X_b), \ldots, g_p(X_b), X_f] \quad (1)$$

if and only if column multiplicity of Karnaugh map $\nu(X_f | X_b) \leq 2^p$ [32,33], where $X_b \cup X_f = \{I_{n-1}, \ldots, I_1, I_0\}$ and $X_b \cap X_f = \varphi$.

Simple serial decomposition leads to the division shown in Figure 1.

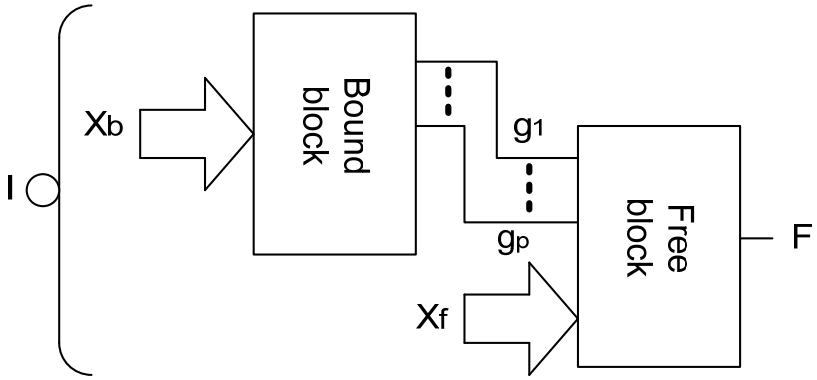

Figure 1. The essence of simple serial decomposition.

The method of describing a function plays a key role in the decomposition process, which determines the computational complexity of individual operations and memory usage. It turns out that in this respect, the representation of the function in the form of BDD [34–36] is very effective. Of course, there are many different forms of BDD. In this paper, the authors refer to the reduced and ordered form of BDD (ROBDD) when using the term BDD.

The essence of BDD decomposition is to cut the diagram horizontally. This divides the nodes into those above the cut line and the nodes below the cut line. The variables associated with the top of the diagram belong to the bound set X_b. On the other hand, the variables associated with the bottom of the diagram belong to the free set X_f. In the case of BDD, column multiplicity corresponds to the number of cut nodes. Cut nodes are the nodes at the bottom of the diagram (below the cut line) to which the edges above the cut line are drawn. The essence of the implementation of decomposition with the use of BDD is shown in Figure 2.

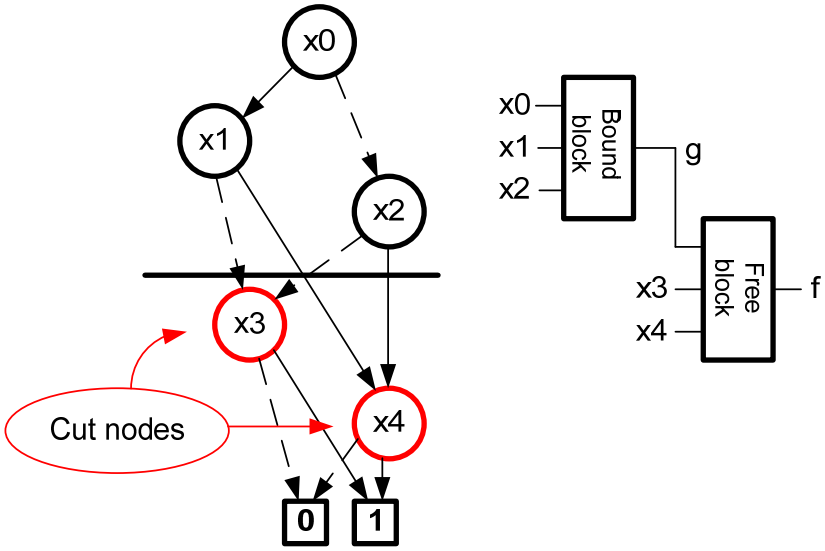

Figure 2. Performing decomposition using BDD.

There are also some modifications to the presented method. As shown in the works [37,38], the decomposition can also be realized by introducing several cut lines for one BDD. Simple serial decomposition is the basis for more complex decomposition models, such as multiple or iterative decomposition, which can also be realized with the use of BDD [39,40].

In order to obtain efficient decomposition, the aim is to limit the number of bound functions, which allows the complexity of the bound block to be reduced. It is, therefore, necessary to limit the number of cut nodes. It turns out that the number of cut nodes depends on two factors: the level of BDD cut and the order of variables in the diagram. The cut level determines the cardinality of the bound set $card(X_b)$. This value corresponds to the number of inputs in LUT block (k). Thus, the cut level is assumed to be k from the root of the BDD $(card(X_b) = k)$. The number of cut nodes at a given level depends on the order of the variables. Therefore, it is necessary to analyze many orderings of variables. The idea of a quick reordering in BDD in terms of decomposition was presented in [41].

As the search for the best decomposition is an iterative process, it is necessary to estimate the effectiveness of the divisions obtained each time. As a rule, the number of LUT block inputs is somewhat configurable, which complicates the process. One of the solutions to this problem is the use of triangle tables [42]. Assuming that the number of inputs is not configurable, instead of determining the technology mapping coefficient from triangle tables, only the cost of implementing the considered decomposition for a constant value of k can be determined.

An example of technology mapping in LUT blocks with three inputs is shown in Figure 3. The xor function described in BDD from Figure 3a was decomposed by a single cut at level 3 from the root. Two cut nodes were obtained, which indicates the necessity to use a single bound function g0. In Figure 3b, the top of the diagram is replaced with a single node associated with the newly created bound function (g0 = x0 xor x1 xor x2). The first step of decomposition ends with reordering the variables, leading to the diagram shown in Figure 3c. In the second step of decomposition, the entire procedure is repeated, replacing the top of the diagram with a single node associated with g1 (g1 = x3 xor x4 xor x5). As a result, the diagram shown in Figure 3d is created, describing the free function (y = g0 xor g1). After implementation in LUT blocks with three inputs, a structure is created as in Figure 3e, the model of which is multiple decomposition.

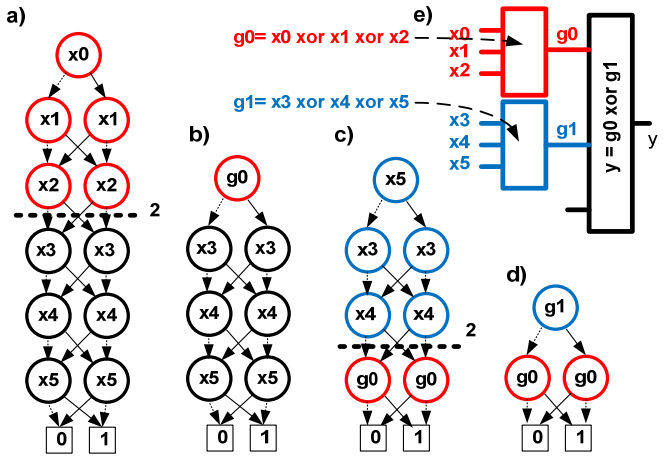

Figure 3. Technology mapping of the xor function in three inputs LUTs: BDD in the next steps of decomposition (**a**–**d**), the result of technology mapping (**e**). The ideas presented may be, depending on the strategy, aimed at minimizing the number of blocks or minimizing the number of logical levels. The main drawback of these approaches is the complete disregard of power consumption.

3. A Synthesis Method Aimed at Minimizing Power Consumption

Power in digital systems can be divided into static power *Pstat* and dynamic power *Pdyn* (2):

$$Pdev = Pstat + Pdyn \tag{2}$$

The static power *Pstat* depends on the technology of making the digital circuit. This power depends on many factors such as: gate leakage, drain junction leakage, sub-threshold current, i.e., a number of technological parameters. In the case of FPGAs, the designer has minimal possibilities to limit static power consumption in the process of designing the implemented circuit. It can only look for an FPGA chip that will ensure minimal static power consumption. In the case of dynamic power *Pdyn*, the designer has a much greater impact on reducing power dissipation. The dynamic power is expressed with the relationship (3):

$$P_{dyn} = \frac{1}{2} V_{dd}^2 f \sum_{i=1}^{n} C_i SW_i \tag{3}$$

It is clearly visible that dynamic power depends on a number of factors. The main factor that allows the dynamic power to be limited is to reduce the supply voltage *Vdd*. Unfortunately, it is practically impossible to limit the supply voltage in FPGAs. Another factor influencing the dynamic power is the frequency of the clock signal *f*. Naturally, it is possible to imagine a local decrease in the clock frequency, although it should be remembered that, as a rule, the aim is to design structures as fast as possible, which means that this method usually cannot be used. The implemented logic structure resulting from decomposition is usually a logic network composed in the general case of n nodes. In the case of FPGAs, the nodes of this network correspond to the LUTs. The internal capacity *Ci* can be associated with the lines corresponding to the connections between the nodes of the network. This capacity must be reloaded each time in the process of switching logic states, which, of course, has an impact on dynamic power consumption. The value of the capacitance *Ci* can be treated individually due to the same technology of making the entire FPGA matrix. Therefore, it becomes crucial to determine how often this capacity is reloaded, i.e., the state changes to the opposite one in the *i*-th node. It is determined by introducing an additional *SWi* parameter into Formula (3), i.e., switching activity for each node. Thus, it can be concluded that limiting the switching activity for as many nodes of the logic network as possible should lead to the limitation of dynamic power dissipation. Thus, the idea of limiting switching activity appears at the stage of designing the implemented structure. However, a problem arises as to how to take into account the

limitations of *SW* in low-level stages of synthesis, such as decomposition or technological mapping. In order to solve this problem, it becomes necessary to develop an appropriate form of description of a logic function, taking into account the parameter expressions in the description.

3.1. Switching Activity

Switching activity determines the probability of changing the state of the considered variable from the value 0 to the value 1 or from the value 1 to the value 0, which is related to the reload of the capacity Ci. Thus, the value of SW can be represented by the Formula (4):

$$SW = P(f_{current} = 0)P(f_{previous} = 1) + P(f_{current} = 1)\ P\left(f_{previous} = 0\right) \quad (4)$$

where P is the probability of the value 1 (or 0) occurring for the considered logic network node at a given time instant. It turns out that the expression (4) can be reduced if, as $P(x)$, we denote the probability of accepting the value 1 by the variable x and all inputs are mutually independent and uncorrelated in time, then the expression (4) can be reduced to the form (5):

$$SW = 2P(x)(1 - P(x)) \quad (5)$$

In this situation, the expression $(1 - P(x))$ is the probability of the value 0. From the perspective of the function variables, it is very difficult to determine the probability value of the value 1. It is difficult to say whether 1 or 0 will occur on the selected input of the combinational circuit and with what probability. In this situation, it is necessary to make certain assumptions as to the probability of the value 1 (most often it is assumed that $P(x = 0) = P(x = 1) = 0.5$).

The relationship (5) shows that the probability of the occurrence of the value 1 in a given network node is related to its switching activity. The highest switching activity occurs for nodes for which the probability of occurrence of the value 1 is 0.5. In the process of synthesizing logic circuits implemented in FPGAs aimed at minimizing power consumption, decomposition becomes necessary, which is the basis for dividing the designed circuit into a network of LUTs. Therefore, there is a need for an appropriate BDD cut that also depends on the SW.

3.2. SWBDD—Switch Activity Binary Decision Diagram

In order to develop a synthesis strategy aimed at minimizing power dissipation, it is necessary to introduce a new diagram form—SWBDD. The new form of the diagram differs from the classic form of BDD (ROBDD) in that additional information has been placed in each node. In a single node, in addition to the information with which the variable is associated, there is also the value of the probability of one $P(fx)$ (to calculate which set of paths in the BDD leading to the leaf "1" will be used) and the SW value determined from the dependence (5) for the considered $P(fx)$. It should be emphasized that when analyzing the paths, it becomes necessary to determine the probability of the occurrence of the value 1 for individual variables. Assuming that the probabilities of the value 1 for all variables are the same and equal to 0.5, an exemplary SWBDD diagram is shown in Figure 4.

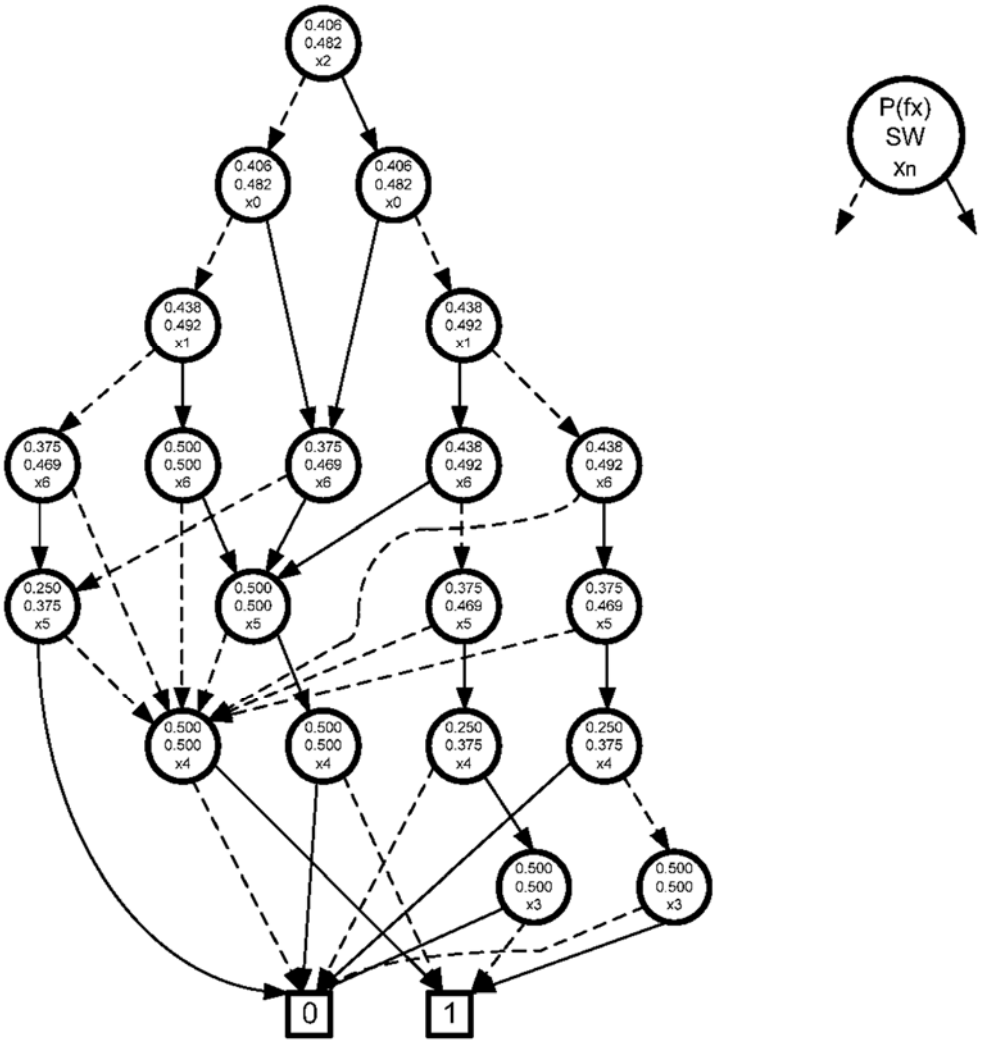

Figure 4. SWBDD diagram.

Assuming the implementation of the function on LUT blocks, it becomes necessary to divide the diagram by cutting the diagram horizontally. It turns out that you can associate the value of the *SW* coefficient with the cut line equal to the sum of the *SWi* values associated with the vertices above the cut line. The value of this coefficient depends on the order of the nodes in the SWBDD.

3.3. Example

Consider the function *f(x0, x1, x2, x3)* described by SWBDD shown in Figure 5. Let us assume that we want to implement it on LUT3/2 blocks (number of inputs/number of outputs). In this situation, it becomes necessary to intersect the SWBDD diagram in the place indicated in Figure 5. In this case, the cardinality of the associated set is 3. For each diagram, the number of cut nodes is the same and includes the node associated with the corresponding variable below the cut line and two leaves. In this situation, in both cases, we need two LUT3/2 blocks to perform the function *f(x0, x1, x2, x3).* When looking at the SW coefficients associated with the nodes above the cut line of the two diagrams, they assume different values. It turns out that the value of the total coefficient for nodes above the cut line for the diagram in Figure 5a is *SWsum* = 1.3359 and is clearly smaller than the value of *SWsum* for Figure 5b (*SWsum* = 2.2109). The sum value of the *SW* coefficients associated with the nodes above the cut line should be used in the process of selecting the appropriate cut line for the SWBDD diagram.

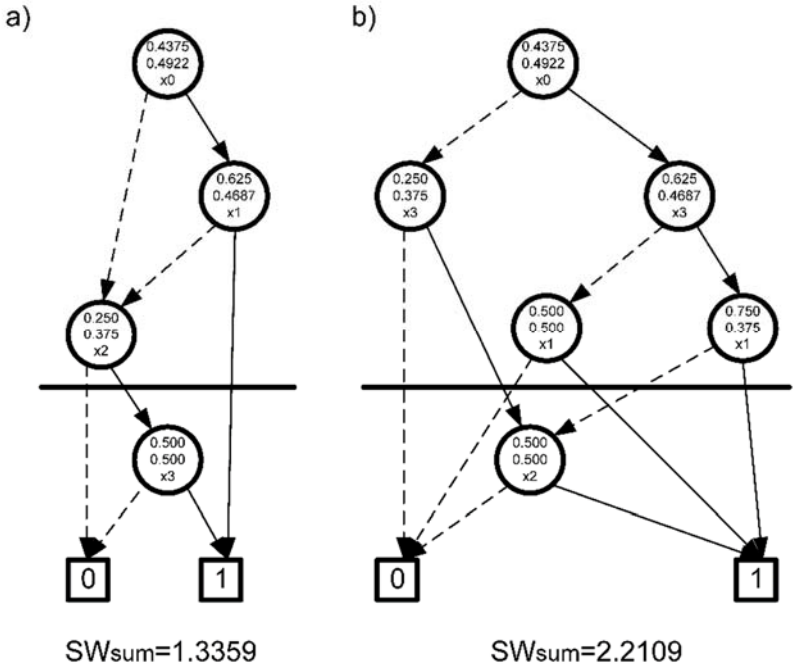

Figure 5. SWBDD for different variable orderings.

4. The Strategy of Low Power Synthesis

As in the case of the classic form of BDD, the form of SWBDD can be used to perform decomposition using a horizontal cut line. As usual, the number of bound functions (*numb_of_g*) and the cardinality of the bound set (*card*(*Xb*)) are key. Of course, as before, some optimization methods can be used, such as non-disjoint decomposition [41,42]. The essence of non-disjoint decomposition is to replace a number of bound functions with input variables, which leads to a reduction in the bound block. Naturally, these variables must meet certain conditions to be able to function as g functions. The number of replaced functions linking the variables x, for non-disjoint decomposition, will be given as *num_ndisj*. It should be emphasized that a reduction in the number of LUTs achieved through non-disjoint decomposition is also beneficial in terms of power consumption reduction.

As in the case of the classic form of BDD, the process of finding the best decomposition in the case of SWBDD is also an iterative process, in which it is necessary to determine the value of the cost function for the implementation of a given cost decomposition. The use of SWBDD, thanks to the additional information contained in the nodes of the diagram, allows for the determination of additional parameters. Let *bound_activity_sum* (*SWsum*) be the cumulative SW value for all nodes above the cut line.

The performed numerous experiments allowed us to determine the cost function, which can be determined from the dependence (6)

$$cost = ((numb\_of\_g - card(Xb)) * 32 + num\_ndisj) * 128 - bound\_activity\_sum \quad (6)$$

The additional constants 32 and 128 were introduced only so that the multiplied part of the expression always has a greater effect on the cost value than the remainder. When analyzing the cost function, it can be observed that *num_ndisj* is always less than 32 and *bound_activity_sum* is always less than 128.

Including *SW* in the function of decomposition cost causes the parameter associated with dynamic power to be added to the assessment of decomposition efficiency. In this situation, it is possible to propose an algorithm of a new synthesis strategy aimed at limiting the dynamic power consumption in the obtained solutions. The algorithm called PowerdekBDD is an extension of the dekBDD algorithm [39,40] known from the literature, which takes into account the element of minimizing dynamic power losses in the decomposition process.

Algorithm of Logic Decomposition Oriented to Power Optimization

The decomposition algorithm consists of the following steps repeated cyclically:

1. Search for the best bound set in a given step (lowest cost function value).
2. Create a g function for a bound block.
3. Create a function diagram with the top of the diagram (above the cut line) replaced by the variables g.
4. If there are still more variables in the diagram than the number of LUT inputs, go to step 1.

The key element during decomposition is finding the appropriate bound set (point 1). Since the bound set corresponds to the part above the cut line in the BDD diagram, searching for the appropriate bound set comes down to changing the order of variables in the diagram at a specific cut level. Checking all order combinations is very time-consuming and has a complexity of $O(n!/(N-k)!)$. Such complexity is unacceptable in practical applications. In the proposed solution, thanks to the heuristic approach, the complexity was limited to $O(k*(n-k))$. The pseudocode of the algorithm for searching for the best set of variables related to a given decomposition step is presented in Algorithm 1. The algorithm's parameters are: F function diagram, k number of LUT block inputs, and n the number of function variables. The order of variables is changed in a double for loop (lines 5, 6). After changing the order of the variables xi and xj (line 7) for the obtained diagram, the cost is determined, allowing the quality of the obtained solution to be assessed.

In order to save computation time, the computation of the cost function was split into two steps: computing the cost related to the number of blockcost_Ftmp blocks (line 8) and computing activity_sum (line 10). The variable activity_sum is the summed activity values for functions rooted in nodes above the cut line. The calculation of activity_sum only takes place if blockcost_Ftmp does not have a greater value than the best solution found so far, thus saving processing time. If a better solution is found, it is remembered (line 13).

Algorithm 1. The Algorithm for Searching for the Best Set of Variables for Bound Set.

```
FindBestBound(F)
2.  //F function
3.  //k number of LUT inputs
4.  //n number of variables
5.  for(i = 0; i < k; i++)
6.    for(j = k; j < n; j++)
7.      Ftmp= Swap(xi, xj, F) //swap variable order in BDD
8.      blockcost_Ftmp= ((numb_of_g − card(Xb)) ∗ 32 + num_ndisj
9.      if(blockcost_Ftmp is lower or equal than before)
10.       activity_sum= CalculateBoundActivitySum();
11.       totalcost_Ftmp= blockcost_Ftmp*128-activity_sum
12.       if(totalcost_Ftmp is lower)
13.         F = Ftmp
```

5. Experimental Results

In order to test the effectiveness of the solutions obtained using the PowerDekBDD algorithm, a number of experiments were performed. The PowerDekBDD algorithm is implemented in the tool of the same name. A popular set of benchmarks [43] was used for the experiments. Since Equation (6) does not take into account the sharing of logic resources between individual functions of the team, it was decided to consider each function separately and then sum the obtained results.

In the first series of experiments, it was decided to compare PowerDekBDD with other academic tools. Since PowerDekBDD is an extension of the DekBDD algorithm, it is crucial to compare both algorithms. Additionally, it was decided to compare with the leading synthesis tool ABC [27]. The work [44] discusses the appropriate scripts for the ABC system: "Baseline" as the basic one, and "PowerMap" and "PowerDC" as the

scripts aimed at reducing power consumption. The comparisons between the tools were made, taking into account the necessary number of LUTs in the obtained "LUTs" solution (assumed k = 5) and the number of logic levels, "Levels," obtained. Of course, additional information has also been added: in the case of the DekBDD and PowerDekBDD tools, the value of the achieved switching activity, "SW," for each benchmark, while in the case of the ABC tool, the value of "Power" was given, which, as stated in [44], is calculated on the basis of "SW." According to the authors, the calculation methods of "Power" and "SW" in both tools are slightly different, and it is difficult to make a direct comparison between them. In order to maintain the reliability of the results, the authors provide the "Power" value for ABC, but at the same time emphasize that the purpose of compiling these results is not to compare them directly. The described list of results is presented in Table 1.

Table 1. Comparison of academic synthesis tools.

Benchm.	Inputs	Outputs	dekBDD			PowerDekBDD			ABC Baseline			ABC PowerMap			ABC PowerDC		
			LUTs	Levels	SW	LUTs	Levels	SW	LUTs	Levels	Power	LUTs	Levels	Power	LUTs	Levels	Power
5xp1	7	10	18	2	7.20	18	2	7.48	27	3	59.19	33	3	65.38	25	3	52.01
9sym	9	1	6	3	2.77	6	3	2.77	63	5	124.34	70	5	132.52	63	5	119.78
alu2	10	8	23	3	9.28	25	4	9.48	28	3	59.78	27	3	56.15	28	3	55.82
clip	9	5	22	3	9.48	24	4	10.16	47	3	98.09	53	3	101.39	45	3	93.09
con1	7	2	3	2	1.40	3	2	1.40	3	2	7.73	3	2	7.34	3	2	7.34
ex1010	10	10	633	8	297.79	631	8	295.60	539	5	1028.28	558	5	1046.18	545	5	1008.31
f51m	8	8	14	3	6.09	14	3	5.90	28	3	60.72	35	3	70.23	26	3	52.66
inc	7	9	18	2	7.40	18	2	7.11	23	3	50.52	27	3	55.25	23	3	49.38
majority	5	1	1	1	0.45	1	1	0.45	1	1	2.92	1	1	2.92	1	1	2.92
misex1	8	7	17	2	6.15	17	2	6.06	17	2	35.79	18	2	35.71	17	2	34.47
mux	21	1	21	6	9.00	17	7	7.46	10	3	24.26	13	3	25.29	11	3	24.64
pcle	19	9	28	3	9.29	25	3	7.67	25	2	49.07	28	2	51.21	25	2	49.02
rd53	5	3	3	1	1.27	3	1	1.27	3	1	8.6	3	1	8.6	3	1	8.6
rd73	7	3	8	2	3.97	8	2	3.97	19	3	41.23	26	4	51.92	19	3	41.18
rd84	8	4	12	3	4.79	12	3	4.79	51	4	105.21	59	5	117.09	52	4	103.47
sao2	10	4	27	4	10.01	25	4	10.26	39	3	83.25	42	3	80.78	40	3	81.47
squar5	5	8	8	1	3.49	8	1	3.49	8	1	20.18	8	1	20.18	8	1	20.18
t481	16	1	5	3	2.12	5	4	2.12	20	4	31.05	19	4	29.72	18	4	27.24
table3	14	14	840	14	338.20	815	17	301.14	698	6	1244.26	724	6	1238.34	698	6	1195.79
term1	34	10	50	5	14.11	48	5	10.51	62	4	124.33	66	4	130.92	66	4	121.22
vg2	25	8	59	5	24.78	60	7	25.21	78	4	141.9	71	4	129.98	77	4	140.66
x2	10	7	14	3	4.07	14	3	3.69	14	2	29.69	15	2	30.24	14	2	29.14
Sum:			1830	79	773.09	1797	88	727.98	1803	67	3430.39	1899	69	3487.34	1807	67	3318.39
Geometric mean:				2.89	7.17	17.46	3.10	6.91	25.46	2.73	54.61	27.63	2.80	56.27	25.35	2.73	52.64

The last rows of Table 1 contain the sum value and the geometric sum. The values of the respective sums are summarized in the graphs in Figure 6.

Analyzing the graph in Figure 6a, it can be observed that the solutions obtained with the PowerDekBDD tool are the most effective in terms of the number of LUTs. The obtained improvement over DekBDD is about 2%. Taking into account Figure 6b, it can be observed that the total number of logic levels for PowerDekBDD is slightly greater than for DekBDD. However, both solutions are much worse than the solutions obtained for ABC. Figure 6c shows the comparison in terms of SW. In the case of PowerDekBDD, this parameter was reduced by over 5% compared with DekBDD. Unfortunately, it is impossible to reliably compare the PowerDekBDD method with the ABC system because the methodology of SW determination in both systems is slightly different; therefore, this issue will not be further developed.

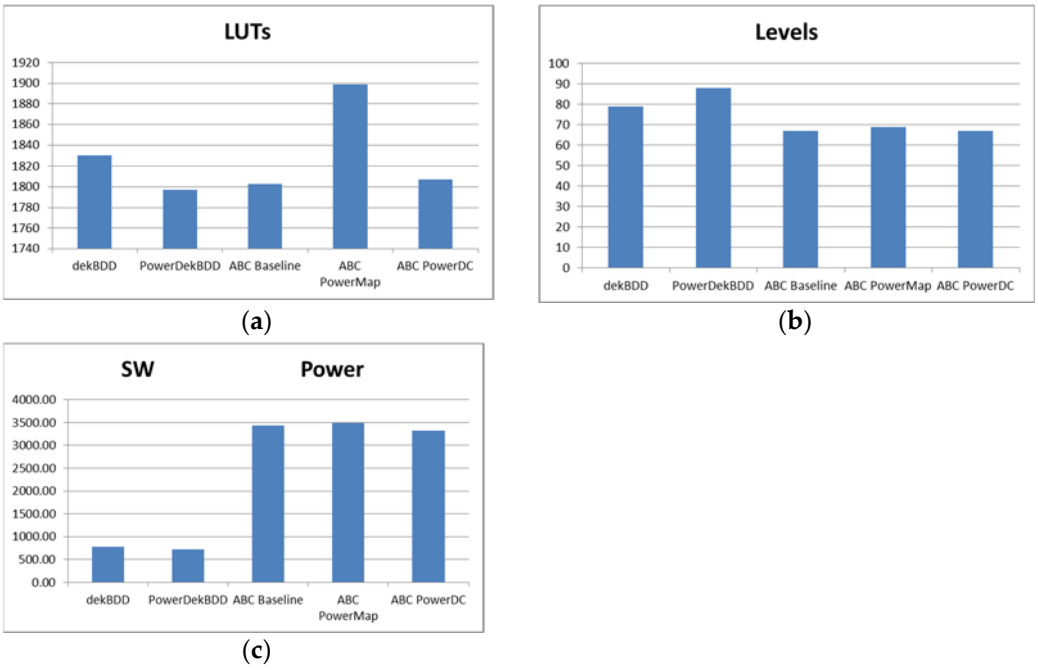

Figure 6. Comparison of academic systems in terms of: (**a**) the number of LUTs; (**b**) the number of logic levels; (**c**) switching activity (Power).

Summarizing the comparison of academic systems, it can be stated that it was possible to reduce the number of blocks, but it is difficult to draw any conclusions as to the reduction in dynamic power dissipation (SW reduction), which is the main subject of this paper. In this situation, it was decided to conduct a second series of experiments, this time using a commercial tool from Xilinx–Vivado [45].

The next steps of the synthesis in the Vivado system were performed with the default settings. The series 7 FPGA (Artix-7) was selected as the target device. The relevant results included in this paper were taken from the reports generated in this tool after the implementation process.

In the PowerDekBDD system, synthesized HDL (Verilog) descriptions of the function after decomposition were generated. Similarly, in ABC, similar descriptions were generated for the PowerDC script (PowerDC was selected due to the fact that the total value of "Power" was the lowest in Table 1). It was decided to perform two series of experiments for PowerDC. In the first "PowerDC SF," HDL descriptions of single functions (without sharing logical resources) obtained from ABC were further synthesized in Vivado. In the second "PowerDC MF," descriptions of sets of functions were further synthesized. PowerDekBDD decomposes individual functions, but providing results for competing systems, including a set of functions, will, according to the authors, better define how the obtained solutions fit into the spectrum of other results. The obtained results are summarized in Table 2. The individual columns in Table 2, for each of the systems, describe: the number of "LUTs" blocks, the dynamic power consumed by the "Dynamic" implementation, and the dynamic power consumed only by the "LUT Power" LUTs.

Table 2. The results of the experiments obtained with the use of the Vivado tool.

Benchm.	Inputs	Outputs	PowerdekBDD SF			PowerDC SF			PowerDC MF		
			LUTs	Dynamic	LUT Power	LUTs	Dynamic	LUT Power	LUTs	Dynamic	LUT Power
5xp1	7	10	11	4.7	0.054	14	4.791	0.045	11	4.486	0.05
9sym	9	1	4	0.869	0.04	6	0.443	0.023	6	0.441	0.023
alu2	10	8	14	2.635	0.057	17	3.002	0.059	19	2.496	0.067
clip	9	5	14	2.929	0.09	25	2.901	0.109	18	2.492	0.084
con1	7	2	2	0.799	0.007	2	0.823	0.008	2	0.799	0.007
ex1010	10	10	170	8.014	0.752	170	8.311	0.736	170	4.797	0.673
f51m	8	8	7	4.108	0.045	12	4.124	0.042	10	3.966	0.04
inc	7	9	9	2.991	0.038	11	3.242	0.035	12	2.848	0.043
majority	5	1	1	0.334	0.003	1	0.333	0.003	1	0.333	0.003
misex1	8	7	8	2.728	0.034	9	2.798	0.032	8	2.664	0.033
mux	21	1	6	0.543	0.025	5	0.54	0.021	5	0.544	0.022
pcle	19	9	12	2.59	0.043	18	2.984	0.051	10	2.551	0.039
rd53	5	3	2	1.492	0.012	3	1.552	0.012	2	1.492	0.013
rd73	7	3	6	1.96	0.029	6	2.043	0.029	6	1.953	0.029
rd84	8	4	7	2.412	0.056	12	2.248	0.057	11	2.095	0.056
sao2	10	4	13	1.63	0.089	20	1.001	0.066	20	0.813	0.071
squar5	5	8	4	2.787	0.026	8	2.962	0.024	4	2.786	0.025
t481	16	1	5	0.717	0.019	7	0.715	0.018	7	0.715	0.018
table3	14	14	270	3.047	0.639	423	5.659	1.349	277	3.102	0.677
term1	34	10	26	2.034	0.1	38	2.711	0.121	16	1.949	0.057
vg2	25	8	26	2.242	0.098	43	2.884	0.143	26	2.239	0.098
x2	10	7	9	1.642	0.036	11	1.802	0.029	8	1.71	0.035
Sum:			626	53.203	2.292	861	57.869	3.012	649	47.271	2.163
Geometric mean:			9.572	1.918	0.045	12.826	1.971	0.045	10.427	1.719	0.042

The last rows of Table 2 contain the sum value and the geometric sum. The values of the respective sums are summarized in the graphs in Figure 7.

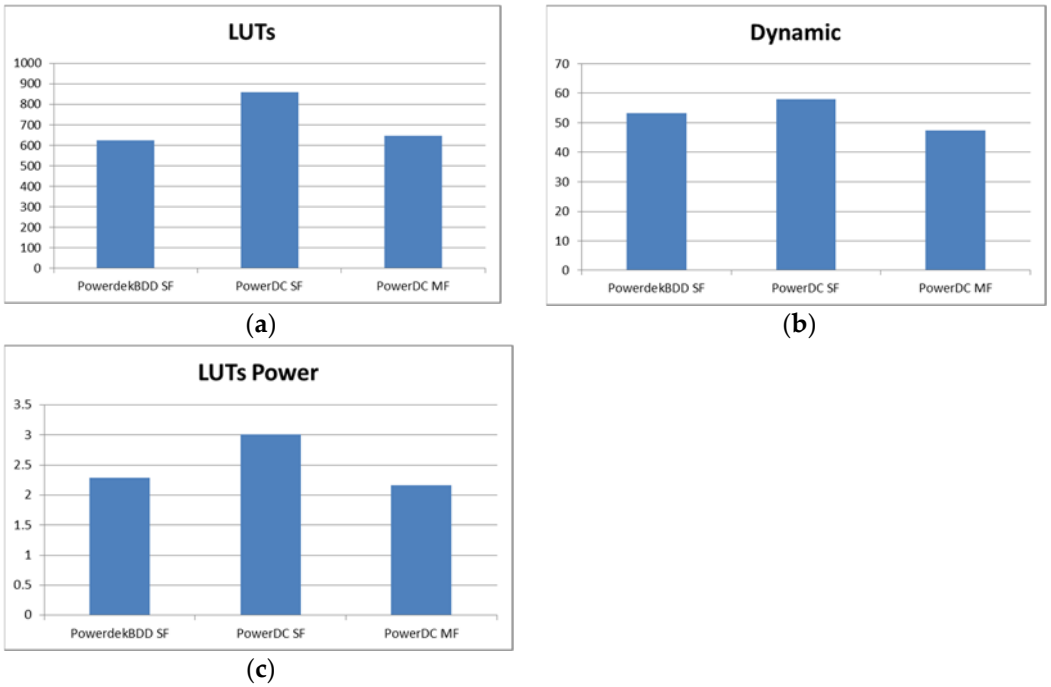

Figure 7. Comparison of the results obtained via Vivado in terms of: (**a**) number of LUT blocks; (**b**) dynamic power consumption; (**c**) power consumption by LUTs.

When analyzing the diagram in Figure 7a, it can be observed that the solutions obtained for PowerDekBDD are very effective in terms of the number of blocks. The obtained results are much better than the results for PowerDC for single functions and practically the same as for the group of functions (thus taking into account resource sharing). Taking into account the overall dynamic power (graph in Figure 7b), it can be observed that the obtained results are slightly worse than for PowerDC MF (which should be expected) and slightly better than for PowerDC (a reduction of 8% was achieved). A very similar situation occurs in the case of the diagram in Figure 7c, showing the dynamic power consumed by the LUTs. In this case, it was possible to achieve a reduction of almost 24% compared with PowerDC SF. It should be emphasized, however, that in both cases, the values of the geometric mean were the same.

6. Conclusions

The new concept of the description of logic functions by means of SWBDD presented in the article is an interesting attempt to include in the process of automatic logic synthesis the issues of power dissipation minimization. It can be a form of description that allows for the minimization of power dissipation in the combination circuits described with the BDD. It can complement many synthesis methods in which BDD is used. This makes it possible to extend known synthesis methods without having to create new synthesis algorithms from scratch.

When analyzing the results of the experiments, it is clear that it is possible to develop a technology representation of the function in the form of a network of LUT blocks, in which there are the smallest possible losses related to the minimization of switching in individual nodes, directly related to the dynamic power dissipated in the circuit. The proposed algorithm is competitive in relation to alternative academic and commercial solutions. A significant weakness of the proposed solution is its limitation related to not taking into account the sharing of logical resources. In its current form, it is possible to analyze each function separately, which means that the obtained solutions can be definitely optimized. This type of optimization will become the basis for further work, the essence of which comes down to the optimization in terms of energy consumption of various synthesis strategies, in which various concepts of structure optimization, the use of a flexible configuration of LUTs, etc. play a leading role.

Author Contributions: Conceptualization, M.K., A.O., and D.K.; methodology, M.K., A.O., and D.K.; software, M.K., A.O., and D.K.; validation, M.K., A.O., and D.K.; formal analysis, M.K., A.O., and D.K.; investigation, M.K., A.O., and D.K.; resources, M.K., A.O., and D.K.; data curation, M.K., A.O., and D.K.; writing M.K., A.O., and D.K., writing—review and editing, M.K., A.O., and D.K.; visualization, M.K., A.O., and D.K.; supervision, M.K., A.O., and D.K.; project administration, M.K., A.O., and D.K.; funding acquisition, M.K., A.O., and D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Raghunathan, A.; Jha, N.K.; Dey, S. *High-Level Power Analysis and Optimization*; Springer: Boston, MA, USA, 1998; ISBN 978-1-4615-5433-2.
2. Benini, L.; De Micheli, G. System-Level Power Optimization: Techniques and Tools. *ACM Trans. Des. Autom. Electron. Syst.* **2000**, *5*, 115–192. [CrossRef]

3. Bondade, R. Hardware-software co-design of an embedded power management module with adaptive on-chip power processing schemes. In Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS), Paris, France, 30 May–2 June 2010; pp. 617–620.
4. Hsiung, P.A.; Liu, C.W. Exploiting Hardware and Software Low power Techniques for Energy Efficient Co-Scheduling in Dynamically Reconfigurable Systems. In Proceedings of the International Conference on Field Programmable Logic and Applications, Amsterdam, The Netherlands, 27–29 August 2007; pp. 165–170.
5. Unsal, O.S.; Koren, I. System-level power-aware design techniques in real-time systems. *Proc. IEEE* **2003**, *91*, 1055–1069. [CrossRef]
6. Wadekar, S.; Parker, A. Interconnect-based system-level energy and power prediction to guide architecture exploration. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *12*, 373–380. [CrossRef]
7. Ali, H.; Al-Hashimi, B.M. Architecture Level Power-Performance Tradeoffs for Pipelined Designs. *IEEE Int. Symp. Circuits Syst.* **2007**, 1791–1794.
8. Bard, S.; Rafla, N.I. Reducing power consumption in FPGAs by pipelining. In Proceedings of the 51st Midwest Symposium on Circuits and Systems, Knoxville, TN, USA, 10–13 August2008; pp. 173–176.
9. Brooks, D.; Tiwari, V.; Martonosi, M. Wattch: A framework for architectural-level power analysis and optimizations. In Proceedings of the 27th International Symposium on Computer Architecture, Vancouver, BC, Canada, 14 June 2000; pp. 83–94.
10. Sunghwan, K.; Jihong, K. Low-power data representation. *Electron. Lett.* **2000**, *36*, 958–959.
11. Chen, C.; Srivastava, A.; Sarrafzadeh, M. On gate level power optimization using dual supply voltages. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2001**, *9*, 616–629. [CrossRef]
12. Manzak, A.; Chakrabarti, C. A low power scheduling scheme with resources operating at multiple voltages. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2002**, *10*, 6–14. [CrossRef]
13. Shin, Y.; Seomum, J.; Choi, K.; Sakurai, T. Power gating: Circuits, design methodologies, and best practice for standard-cell VLSI designs. *ACM Trans. Design Autom. Electron. Syst.* **2010**, *15*, 1–37. [CrossRef]
14. Wang, N.; Zhong, W.; Chen, S.; Ma, Z.; Ling, X.; Zhu, Y. Power-gating-aware scheduling with effective hardware resources optimization. *Integr. VLSI J.* **2018**, *61*, 167–177. [CrossRef]
15. Kuc, M.; Sulek, W.; Kania, D. Low Power QC-LDPC Decoder Based on Token Ring Architecture. *Energies* **2020**, *13*, 6310. [CrossRef]
16. Alidina, M.; Monteiro, J.; Devadas, S.; Ghosh, M.A. Papaefthymiou, Precomputation-based sequential logic optimization for low power. *Ieee Trans. Very Large Scale Integr. (VLSI) Syst.* **1994**, *2*, 426–436. [CrossRef]
17. Benini, L.; De Micheli, G.; Macii, E.; Odasso, G.; Poncino, M. Kernel-based power optimization of RTL components: Exact and approximate extraction algorithms. In Proceedings of the 36th annual ACM/IEEE Design Automation Conference, New Orleans, LA, USA, 21–25 June 1999; pp. 247–252.
18. Benini, L.; De Micheli, G.; Lioy, A.; Macii, E.; Odasso, G.; Poncino, M. Synthesis of power-managed sequential components based on computational kernel extraction. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2001**, *20*, 1118–1131. [CrossRef]
19. Sudnitson, A. Computational kernel extraction for synthesis of power-managed sequential components. In Proceedings of the IEEE 9th International Conference on Electronics, Circuits and Systems, Dubrovnik, Croatia, 15–18 September 2002; Volume 2, pp. 749–752.
20. Chow, S.-H.; Ho, Y.-C.; Hwang, T. Low Power Realization of Finite State Machines A Decomposition Approach. *ACM Trans. Des. Autom. Electron. Syst.* **1996**, *1*, 315–340. [CrossRef]
21. Monteiro, J.; Oliveira, A. Finite State Machine Decomposition for Low Power. In Proceedings of the 35th Design Automation Conference, San Francisco, CA, USA, 15–19 June 1998; pp. 758–763.
22. Modrzyk, D.; Kania, D. Asynchroniczna wymiana danych w układzie GALS ukierunkowana na minimalizację poboru mocy. *Prz. Elektrotechniczny* **2014**, *90*, 132–137.
23. Barkalov, A.; Titarenko, L.; Mielcarek, K. Improving characteristic of LUT based Mealey FSMs. *Int. J. Appl. Math. Comput. Sci.* **2020**, *30*, 745–759.
24. Barkalov, A.; Titarenko, L.; Chmielewski, S. Improving Characteristics of LUT-Based Moore FSMs. *IEEE Access* **2020**, *8*, 155306–155318. [CrossRef]
25. Kubica, M.; Kajstura, K.; Kania, D. Logic synthesis of low power FSM dedicated into LUT-based FPGA. In Proceedings of the ICCMSE 2018, American Institute of Physics, Thessaloniki, The Hellenic Republic, 14 March 2018; Volume 2040.
26. Kajstura, K.; Kania, D. Low Power Synthesis of Finite State Machines State Assignment Decomposition Algorithm. *J. Circuits Syst. Comput.* **2018**, *27*, 1850041. [CrossRef]
27. Berkeley Logic Synthesis Group. ABC: A System for Sequential Synthesis and Verification. Available online: http://www.eecs.berkeley.edu/alanmi/abc (accessed on 21 December 2005).
28. Felipe Machado, S. A BDD Proposal for Probabilistic Switching Activity Estimation. 2009. Available online: https://oa.upm.es/3343/ (accessed on 10 March 2020).
29. Lindgren, P.; Kerttu, M.; Thornton, M.; Drechsler, R. Low power optimization technique for BDD mapped circuits. In Proceedings of the ASP-DAC 2001, Asia and South Pacific Design Automation Conference 2001 (Cat. No.01EX455), Yokohama, Japan, 2 February 2001; pp. 615–621. [CrossRef]
30. Balasubramanian, P.; Anantha, K. Power and Delay Optimized Graph Representation for Combinational Logic Circuits. World Academy of Science, Engineering and Technology. *Int. J. Comput. Electr. Autom. Control Inf. Eng.* **2007**, *1*, 2475–2481.

31. Mehrotra, R. Systematic Delay-Driven Power Optimisation and Power-Driven Delay Optimisation of Combinational Circuits. Ph.D. Thesis, University College Cork, Cork, Ireland, 2013.
32. Ashenhurst, R.L. The decomposition of switching functions. In Proceedings of the An International Symposium on the Theory of Switching, Cambridge, MA, USA, 2–5 April 1957.
33. Curtis, H.A. *The Design of switching Circuits*; D. van Nostrand Company, Inc.: Princeton, UK; Toronto, ON, Canada; New York, NY, USA, 1962.
34. Scholl, C. *Functional Decomposition with Application to FPGA Synthesis*; Kluwer Academic Publisher: Boston, MA, USA, 2001.
35. Akers, S.B. Binary Decision Diagrams. *IEEE Trans. Comput.* **1978**, *C-27*, 509–516. [CrossRef]
36. Minato, S. *Binary Decision Diagrams and Applications for VLSI CAD*; Springer: Berlin/Heidelberg, Germany, 1995.
37. Kubica, M.; Kania, D. Area-oriented technology mapping for LUT-based logic blocks. *Int. J. Appl. Math. Comput. Sci.* **2017**, *27*, 207–222. [CrossRef]
38. Kubica, M.; Kania, D. Decomposition of multi-output functions oriented to configurability of logic blocks. *Bull. Pol. Acad. Sci. Tech. Sci.* **2017**, *65*, 317–331. [CrossRef]
39. Opara, A.; Kubica, M.; Kania, D. Strategy of logic synthesis using MTBDD dedicated to FPGA. *Integr. VLSI J.* **2018**, *62*, 142–158. [CrossRef]
40. Kubica, M.; Opara, A.; Kania, D. Logic synthesis for FPGAs based on cutting of BDD. *Microprocess. Microsyst.* **2017**, *52*, 173–187. [CrossRef]
41. Opara, A.; Kubica, M.; Kania, D. Methods of improving time efficiency of decomposition dedicated at FPGA structures and using BDD in the process of cyber-physical synthesis. *IEEE Access* **2019**, *7*, 20619–20631. [CrossRef]
42. Kubica, M.; Kania, D. Technology mapping oriented to adaptive logic modules, Bulletin of the Polish Academy of Sciences. *Tech. Sci.* **2019**, *67*, 947–956.
43. Collaborative Benchmarking and Experimental Algorythmics Laboratory, A Benchmark Set. 2008. Available online: http://www.cbl.ncsu.edu:16080/benchmarks/LGSynth93/testcase/ (accessed on 10 March 2020).
44. Jang, S.; Chung, K.; Mishchenko, A.; Brayton, R.; Xilinx Inc. A Power Optimization Toolbox for Logic Synthesis and Mapping. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.152.2775 (accessed on 10 March 2020).
45. Xilinx. Vivado Design Suite User Guide Implementation. UG904. 2021. Available online: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_4/ug904-vivado-implementation.pdf (accessed on 10 March 2020).